(2007) *Journal of Cognitive Engineering and Decision Making*, 1(2). 121-147.

Including a model of Visual Processing with a Cognitive Architecture to Model a Simple Teleoperation Task

24 February 2007

Frank E. Ritter[2,3], Urmila Kukreja[2], and Robert St. Amant[1]

[1]Department of Computer Science, North Carolina State University
Raleigh, NC 27695 USA

[2]Department of Computer Science and Engineering, The Pennsylvania State University
University Park, PA 16802 USA

[3]College of Information Sciences and Technology, The Pennsylvania State University
University Park, PA 16802 USA

Abstract

This article presents progress in providing user models with sufficient visual information and motor control to perform teleoperation with an unmodified, physically realized robot. User models built by extending cognitive models to interact directly with interfaces can provide a theoretical basis for predicting user behavior. These models can help summarize and explain usability issues in domains in which conventional user testing is too time-consuming, too demanding of other resources, or too dynamic for static models. The user model consists of an ACT-R cognitive model and the SegMan image processing and interpretation system. ACT-R supports directing simple rover navigation and response time predictions. SegMan supports interpreting many aspects of HCI interfaces and can now interpret simple aspects of video used in simple navigation tasks and provide keypresses and mouse actions directly. Processing limited amounts of an image as a human fovea helped make this system work in real time. A study in robot teleoperation provides evidence that the cognitive and perceptual/motor model approximates human behaviour (based on comparison of task time, learning behavior, and mouse actions) in a simple navigation task. This work demonstrates how user modelling techniques are maturing to the extent that they can be used for assessing interfaces for dynamic tasks by predicting performance during teleoperation, a common HRI task.

Introduction

As computer and robotics technology has progressed, systems have simultaneously become more capable, more affordable, and a more common part of the work environments and everyday lives of a wide range of users. Human-robot interaction (HRI) and human-computer interaction (HCI) share a central concern of usability. Several developers of robots for teleoperation have argued that usability is important. They note that operating a robot is a complex task (Casper & Murphy, 2003); understanding how operators use robots will remain a problem for some time, and is of interest to robot developers (Cradall, Nielsen, & Goodrich, 2003; Drury, Hestand, Yanco, & Scholtz, 2004; Marble, Bruemmer, Few, & Dudenhoeffer, 2004; Olsen Jr. & Goodrich, 2003; Scholtz, 2003; Scholtz, Antonishek, & Young, 2003). Several robot developers have gone so far as to argue and present evidence that robots that are not usable will not be used (Murphy, Blitch, & Casper, 2002).

While HCI evaluation techniques are robust and may have clear application to HRI, HRI poses more difficult problems for evaluation because it covers a wider range of tasks and environments. HRI extensions to new tasks and environments open the possibility that novel approaches to evaluation may be useful. There are also other differences between HCI and HRI that make some of the evaluation techniques less applicable. Scholtz (2003) reviews these differences, and we examine them in detail in the next section.

Yanco, Drury, and Scholtz (2004) identify three classes of evaluation methods common in HCI that might be used in HRI: inspection methods carried out by interface design experts; empirical methods based on observation and testing with users; and formal methods based on analytical models of interaction. These classes of evaluation approaches have resulted in significant advances in our understanding of interaction in both HCI and HRI. Inspection methods for improving conventional user interfaces has been documented extensively by Nielsen (1993); the evaluation guidelines provided by Scholtz (2002) and refined by Yanco et al. (2004) can be viewed as a high-level set of inspection methods that provide insight into HRI tasks. Empirical evaluation based on user studies, both in the laboratory and in the field, are staples in HCI and HRI development (e.g., Scholtz, Antonishek, & Young, 2003); descriptions of interface projects are generally considered incomplete without some reference to user testing. Formal mathematical methods, including user models, while less common, have been used to rigorously identify and analyze usability flaws in everyday systems such as calculators (Thimbleby, 2002). We are only aware of limited research on the use of user models for HRI applications. This type of work, on the application of GOMSL models for representing user behavior in a simple rover navigation (Kaber, Wang, & Kim, 2006), and on mobile robots replicating human "hide and seek" tasks (Trafton et al., 2006), has primarily focused on high-level modeling approaches. These methods all influence design both directly through feedback on a specific interface and indirectly by designers modifying their design in anticipation of the feedback.

The focus of this article is on a technique for HRI evaluation based on detailed cognitive modelling. A computational cognitive model is a theory of human cognition that is realized as a computer program. Cognitive modelling provides a way of applying what is known about human behavior both to explain and to reproduce human behavior in specific tasks. Cognitive architectures and models created in them integrate theories of cognition, visual attention, and motor movement. They were initially created as scientific theories to understand cognition (e.g., Newell & Simon, 1972) but they increasingly have been used to evaluate human-computer interaction in a wide variety of application domains (e.g., Byrne & Kirlik, 2005; Gluck, Ball, Krusmark, Rodgers, & Purtee, 2003; Kieras, Wood, & Meyer, 1997; Salvucci, 2001; St. Amant, Horton, & Ritter, in press). Models apply specific problem-solving strategies to perform the tasks of interest, reproducing such aspects of human behavior as time durations for high- and low-level actions, occurrences of errors, and abstract behaviors such as exploration and learning. Several authors have argued that with the help of cognitive models, the design of interactive systems can be approached as a type of engineering design (Byrne & Gray, 2003; Kieras, Wood, & Meyer, 1995; Newell, 1990; Ritter, Baxter, Jones, & Young, 2000; Ritter, Van Rooy, & St. Amant, 2002), but in most cases the models have not been used to directly exercise the interfaces being examined.

Cognitive modeling operates at a more detailed level than these task analysis methods. What benefits might a cognitive model that interacts directly with an interface bring that are not already provided by established techniques for user interface evaluation? Proponents of cognitive modelling can point to a number of potential results that make pursuit of the approach worthwhile. First, cognitive models provide accounts of user behavior that generalize across a variety of environments and tasks because they can perform the tasks in those environments. They not only make predictions of what users will do but provide explanations (in a mechanistic sense) for why the activities unfold in particular ways (e.g., Byrne & Kirlik, 2005; Gluck, Ball, Krusmark, Rodgers, & Purtee, 2003). Second, cognitive models can give such predictions and explanations at a more detailed level than is often possible with other techniques. Because cognitive models can rely on an explicit architecture of cognitive processing, the behavior can be predicted in detail based on what cognitive, perceptual, and motor mechanisms are being used, and explained directly from such factors as dependence on specific visual search strategies, or constraints on the speed of finger movement, or limitations of working memory and retrieval speed (e.g., Gluck & Pew, 2005; Gray, in press; Salvucci, 2001). Knowing and understanding these trade-offs can be important for design (e.g., Byrne & Gray, 2003; Kieras, Wood, & Meyer, 1995).

Third, cognitive models offer new opportunities for experimental evaluation of interactive systems. User testing is expensive, which is exacerbated in HRI by the need to work with physical robots and environments. As models develop, they offer the potential to be less expensive than real users, in the same way that CAD/CAM systems are cheaper than building prototypes. Using a cognitive model as a surrogate user also avoids several potential confounds in evaluation and difficulties that real users present, including experimental confounds due to learning and practice effects, unpredictable user backgrounds and levels of knowledge. Models avoid these problems because they can simply be reinitialized for each new trial, and can be tasked to test every part of an interface. Where it may not be feasible to apply models every time, using models to predict performance will lead to better models of users in interface designers' heads. Finally, cognitive models are dynamic. Where the task is realised in a physical robot, the interactions (tasks and task frequency) may be difficult to specify in detail and difficult to analyse with static descriptions. A user model based on a cognitive architecture can actively react and control the robot. A trace of its behavior reports what tasks were performed and their duration, and to a certain extent, how successful the tasks were (e.g., Ritter, Van Rooy, St. Amant, & Simpson, 2006; Salvucci, 2001).

An important first challenge is determining whether it is feasible to build low-level cognitive models that can carry out HRI tasks as surrogate users working directly with interfaces. In the next section we outline the general ways in which HRI task environments differ from more conventional HCI environments. These differences motivate our focus on a specific aspect of user modelling: providing a flexible and robust way for user models to deal with the complexities of interacting with both virtual and physical environments inherent in HRI, and considering how this leads to a more dynamic form of task analysis. These issues are mainly related to processing of the visual environment.

While many user models work with simulations of interfaces and to good effect (e.g., Kieras, Wood, Abotel, & Hornof, 1995; Kieras, Wood, & Meyer, 1997; Urbas & Leuchter, 2005), and some models interact with instrumented interfaces (some are reviewed in Ritter, Baxter, Jones, & Young, 2000), it can be difficult to create simulations of the robot and also its environment. Interacting directly with a robot's interface will allow models to be more easily deployed and to be more veridical (there is concern in the robot research community about using simulators to replace robots in studies).

We describe a system called SegMan that addresses this problem of interaction. It provides information from the environment to the cognitive model in a form modelling theories of intermediate human visual processing and provides outputs from the model to the environment. SegMan is informally evaluated here in the context of a robot navigation task. Our results suggest that cognitive modeling techniques have matured to the point that they can start to be used for evaluation of some tasks in HRI.

Cognitive Models for Evaluating Interactive Systems

To evaluate a user interface using a cognitive model, the following approach is typical. Based on pilot studies and the psychological literature, a researcher will develop a cognitive model for one or more tasks supported by an interactive application. For most cognitive architectures this entails encoding relevant declarative task knowledge as memory chunks, building productions that represent the processing of this knowledge, and providing for the model's interaction with a task environment in the form of appropriate input of information and output of actions. The model is incrementally refined until it satisfies the researcher's requirements with respect to detail and accuracy, given preliminary data from pilot studies and the literature. An experiment is then designed in which human participants carry out the same tasks in the interface that the model performs. Data is collected for comparison between human and model performance. Statistical testing then serves to validate or disconfirm the model in its representation of human performance and to suggest improvements. Then, the model can be applied or the cycle of model improvement and testing can be repeated. Further details on this approach and the work that inspired our view are available (Kieras, 1985; Newell & Simon, 1972; Ritter, 2004).

Interactive computing environments vary in the degree to which they are amenable to analysis by existing usability techniques. Research in artificial intelligence provides a general framework for characterizing problem-solving environments (Russell & Norvig, 1995), a framework that can be exploited in describing interactive computing systems. (Scholtz et al. (2004) give an alternative description of the properties of HRI environments, which we discuss in further detail below.) For environments that are fully observable, deterministic, episodic, static, discrete, and single-user, the space of decisions and actions is significantly constrained in comparison with environments with the opposite properties of partially observable, non-deterministic, sequential, dynamic, continuous, and multi-user.

In simple environments it is possible for simple analysis techniques to be applied. In complex environments it becomes more difficult to apply models based on simple assumptions. There are many sources of variability in complex environments. The state of the interface and the pattern and even the number of task actions can vary dynamically across trials, tasks, users, or time. Models can help reduce variance here because they are themselves more repeatable and inspectable.

Cognitive modeling has also been applied to the evaluation of many types of applications that do not match idealized, simple environments like desktop word processing applications. It is possible now to design cognitive models that to a large extent can replicate the problem-solving behavior of human users on tasks of practical interest in complex environments—in particular, those needed to interact with interactive software applications. For example, Salvucci (2001) has carried out a research program to understand the relationship between driving performance and the use of in-car interactive systems. Schoelles and Gray (2001) have done similar work with an air-traffic control-like task, where the environment is stochastic, continuous, and only partially observable. Ritter, Van Rooy, St. Amant, and Simpson (2006) had a model interact with an unmodified task similar to robot driving, but the environment was completely simulated in software and was much more predictable than robot teleoperation.

Implications for modeling in HRI

HRI task environments have a number of properties that make them more complex to analyze than conventional software applications. The key difficulty in modeling HRI tasks arises from the embodiment of robots in physical environments. Consider some of the common requirements on interactions that can be involved in a robot control task:

- A user must carry out a repetitive task in a changing environment, responding in real time to potentially unexpected external events and effects of the robot's actions.
- A user must interpret scenes captured by one or more cameras on a robot displayed in windows on a computer screen; the scenes may not be directly visible to the user or may be visible but from a different viewing angle.

- A user must deal with unfamiliar classes of information from novel (to the user) sensors: infrared, bump, sonar, and so forth. Information fusion may produce difficult-to-interpret presentations due to such unfamiliarity. Problems may arise in the robot's activities, problems that would be easily understood by the user if detectable by sensors but are otherwise effectively invisible.
- A user must integrate information from a physical and a virtual environment that can appear loosely coupled. For example, as a robot turns in a circle, the stationary user monitoring the robot's camera view must keep track of the changing relationship between the robot's orientation and the interface.
- A user must often provide high-level guidance to a robot, leaving lower-level operations to the robot's internal controller. In cases of mixed initiative or adjustable autonomy, the level at which the user must interact with the robot can change without an explicit directive from the user.
- A user must control multiple robots at the same time. The robots may be independent, cooperating, or even to some extent competing against one another (e.g., for space or other resources). The robots may be identical or heterogeneous. The robots may be carrying out different tasks; even if the tasks are similar, different local environment properties may raise differing control issues.

Thus, HRI commonly takes place in task environments that are only partially observable, stochastic, sequential, dynamic, and continuous. This is not the entire story, however. Scholtz et al. (2004) describe six dimensions along which HRI differs from HCI: (a) a wider variety of roles that users and operators play in controlling a robot; (b) the ways that a robot platform interacts with the physical environment is less predictable; (c) the dynamic behavior of the hardware (leading to autonomous changes); (d) the nature of the user’s interaction environment; (e) the number of robots that the user controls; and (f) the autonomous behavior of the robot(s). These dimensions are at a different level of abstraction than the properties used to describe the problem-solving environments above, but can be viewed as partitioning and categorizing sources of differences between HCI and HRI with respect to problem-solving properties. Essentially, the user or operator in an HRI environment must potentially deal with several coupled problem-solving environments:

- Decision making in a human-human collaborative context, with differing roles. This environment may be only partially observable, nondeterministic, sequential, dynamic, and continuous.
- Monitoring of robot hardware. Tasks in this problem-solving environment may be no less complex.
- Monitoring of robot behaviors in the environment. Again, because of physical interactions, this environment can be difficult for problem solving; additional complexity arises in the case of multiple robots.

- Interaction with an HRI control interface. Only in this last case can be the problem-solving environment be reduced, in the best case, to a standard HCI environment.

HRI task environments thus provide a significant challenge for effective interaction and by implication for evaluation by a cognitive model. In the environmental features noted above, two recurring issues are the need to interpret a complex information environment, with most data provided by visual channels, and to manage complexity in controlling the behavior of robots. Much of this complexity must be handled by explicit reasoning in a model about what is observed and what actions are to be taken. For example, in navigation tasks, the positions of objects visible in a camera view over the course of a robot's movement may not be perfectly predictable because of wheel slippage or the presence of unperceived and unexpected obstacles. A more immediate issue, however, is ensuring that relevant information to perform these tasks is accessible to a model in the first place and how visual information is to be transformed into objects with explicit symbolic and numerical properties.

At the time cognitive architectures were created (Newell, 1990), interaction was often done with toy tasks and in non-challenging environments. To interact with a commercial robot's interface will require extending the cognitive architecture to interact with the world—an explicit way for the architecture to apply knowledge to perform the task. Current cognitive architectures are ill-suited for also simulating the details of robots and their environments—the model will have to interact with a real or simulated robot. We next examine a way for cognitive architectures to be expanded to support this more complex application when the robot's interface cannot be modified to support the model's use.

More Direct Visual Processing for a Cognitive Architecture

When evaluating a computer system, either in an HCI or HRI context a cognitive model needs access to the information that a user sees (Ritter, Baxter, Jones, & Young, 2000; Ritter, Van Rooy, & St. Amant, 2002). An open issue is how environmental information can best be translated into visual objects. Historically, cognitive models have interacted with specifications of tasks, with task simulations, or with special-purpose versions of task environments tailored to the needs of a cognitive model, rather than the task environments that users work in (Ritter & Major, 1995). For example, the models in the AMBR project (Gluck & Pew, 2005) required that an API first be developed for data exchange between the simulation and the human performance models. There are two problems with this approach. One is that the task has to be created twice, which requires extra work. The second problem is that the task and the simulated task are not always the same. Models embedded in such environments are often limited by the accuracy and precision of a programmatic interface to the environment, rather than by the factors of the environment itself. Third, the resulting models are tied to an ad hoc interaction architecture, are often less inspectable, and are less often reusable.

For example, currently, ACT-R models typically receive information about a visual environment through specialized functions (Byrne, 2001) that interact with specially instrumented windows. These windows are built in a tool included with ACT-R to create simulated task environments. The resulting simulations have a visual display that human users can interact with as well, and some work has used them directly where the task had to be created. At this level of representation, ACT-R incorporates a plausible, reusable theory of perception and motor behavior. This theory is based on existing theories of attention (Treisman & Gelade, 1980) and some properties of early visual processing such as feature integration borrowing heavily from EPIC (Meyer & Kieras, 1997).

To be fair, there are legitimate trade-offs in this area. Interacting with a simulated interface provides a lot of control about what can be seen and how well it can be seen. For many tasks, duplicating the task in a way that the model can interact with it can be the best way to go. Any fidelity lost in using a task simulation versus the real task represents a trade-off of similar deviations of the SegMan tool from accurate representation of human visual processing at low-levels.

We have pursued an alternative approach that we believe has longer-term generality. Rather than specializing the environment to the needs of a cognitive model (e.g., a cognitive model interaction management system, Ritter et al., 2000), we have worked toward extending the visual processing capabilities of a cognitive modeling architecture to use the environment without modifications. We believe that eventually, if we are to reach the goal of building models that automatically interact with a wide range of real environments, the issue of visual processing by cognitive and user models must be addressed directly, even under the constraint of incomplete scientific knowledge of vision. Thus, we next discuss the SegMan system that allows architectures to interact more directly.

SegMan overview

We have developed an image processing substrate, called SegMan (St. Amant, Horton, & Ritter, 2004; St. Amant & Riedl, 2001), that extends ACT-R's (and other cognitive architectures') visual processing to work with interfaces through parsing their bitmap. SegMan was originally developed to allow user interface agents to control off-the-shelf interactive software applications through their graphical user interfaces. Systems using SegMan with AI techniques have modelled navigating through menus and dialog boxes in the Windows user interface (St. Amant & Riedl, 2001), with Soar and ACT-R have modelled several game-playing tasks (St. Amant, Riedel, Ritter, & Reifers, 2005), with EPIC reproduced a basic psychological experiment (St. Amant, Riedel, Ritter, & Reifers, 2005), and most recently drove a simulated car to generate lessons for HCI and HRI (Ritter, Van Rooy, St. Amant, & Simpson, 2006). Over time, we have attempted to refine SegMan to reflect research in visual processing, in particular models of intermediate vision.

In describing SegMan as a cognitive modeling component, we first discuss its design from a computational perspective, then its correspondences to a current model of intermediate vision. Visual properties in SegMan are of three types: properties of pixel regions, relationships between pixel regions, and composite properties of pixel groups. We refer to these collectively as region/group properties. They build upon each other so we start with the first two that work closely together.

SegMan's most basic representation of visual objects is *pixel regions*: non-overlapping sets of visually adjacent groups of like-colored pixels. The source for pixel regions is the display of a computer screen, which can range from standard interface widgets to real images from a video feed. Figure 1 will be used to show how SegMan works. The schematic diagram in Figure 1 shows four pixel regions: one corresponding to the letter 'F', two corresponding to the letter 'i', and one representing the background. For the purposes of recognition, the most important properties of a pixel region are related to its shape.

The shape properties of a region are encoded as a summary of the connectivity relationships between its constituent pixels. Each pixel *p* in a region has eight neighbouring pixels that may be members of the same region, represented in a 3 x 3 mask overlaid on the position of *p*. In Figure 1, pixel *p* has three neighbours in the same region, at relative positions 2, 4, and 6. For each pixel in a region, SegMan summarizes its neighbor value with the function

$$v(p) = \sum_{i=0}^{7} 2^{is(p,i)}$$

Here i iterates over the positions of all the neighboring pixels in the mask and $s(p,i)$ is a function that returns 1 if p and i are in the same region, 0 otherwise. For pixel p in Figure 1, $v(p) = 2^2 + 2^4 + 2^6 = 84$. A single pixel's value for v can range from 0 (for an isolated pixel with no neighbors in the same group) to 255 (for a pixel surrounded entirely by neighbors in the same region). SegMan represents the shape of entire pixel regions by maintaining an array of length 256, in which an entry at position k contains the number of pixels for which $v(p) = k$. For example, in a pixel region that represents a solid square region five pixels on a side, the inner nine elements all have a pixel neighbor value v of 255; the entry at index 255 in the array for the pixel region is thus 9. For the body of the "i" in Figure 1, the array has mostly elements filled with 0, but the four pixels are represented with elements $4(2^2)$: 1; $64(2^6)$: 1; $68(2^2 + 2^6)$: 2. The dot has 0: 1, and the rest of its cells are set to 0.

< Insert Figure 1 about here. >

Pixel regions can be combined into higher-level patterns called *pixel groups*. For example, a three-dimensional effect for a button in a graphical user interface is achieved by one or more dark outlines below a lighter rectangular shape. These shadow outlines are pixel regions with specific adjacency relationships to the region representing the surface of the button.

Specific combinations of region/group properties can be organized into declarative templates and stored in a library. For example, the shadowed button described above could be specified in a template that defines a size threshold for the pixel region representing the "surface" of the button, the color of the pixel region, and the relationships to adjacent shadow pixel regions. Constraints on region/group properties are treated as propositions and combined in logical formulas. Named templates can be retrieved from the library to identify known objects. Currently, SegMan has templates for three fonts, about 50 types of Windows objects, and several dozen objects and object types it has used in various tasks, including the simple region used in this task to recognize the robot's path. These objects can have an RGB color, which is also used to differentiate regions.

As primitives for representing a visual environment, propositions based on pixel regions have significant limitations. For example, detectable objects must have sharp boundaries against their background and limited variation in color; processing is not adaptive, and the same object may not be identifiable from different visual angles. This means that applying SegMan in realistically complex environments is more of an engineering challenge than a machine vision problem. The pixel-based representation works surprisingly well in practice, however. St. Amant et al. (2005) provide a review of how it has been used in the processing of photographic images of cell phones, the interface to an online gambling casino, and dynamically changing views in a driving game, among other visual domains. Video is displayed as part of the interface in this task, so SegMan can be applied to video by treating the video as a series of static images on the bitmap, which is what the interface displays it as.

To add new objects to the libraries, representative static images are sampled from the target visual environment. A tool in the SegMan development environment processes each image and provides a set of recognizable pixel regions and their potential combination into pixel groups. This set is pruned and tailored by hand for accuracy and generality. The resulting set of patterns is stored in the library for use in the environment by an agent at run time. The entire process is iterative; in the environments with which we have experimented, the process takes only a few iterations to reach adequate performance.

SegMan operation

SegMan generates and maintains its representation in three stages: segmentation, feature computation, and interpretation. In the segmentation stage, a sample of pixels is selected. Sampling may occur at different resolutions (e.g., every pixel or only every tenth pixel). Within specific boundaries, which by default encompass the entire display, the sampled pixels are treated as seeds for expansion into pixel regions, through a bounded depth-first search. In the feature computation stage, properties are computed for each pixel region, regions are combined into pixel groups, and group properties are computed. The third stage, interpretation, involves a matching process between library templates and the results of the earlier stages. Interpretation is top-down, in contrast to segmentation and feature computation.

SegMan's current design is most strongly influenced by Roelfsema's (2005) theory of visual processing, an elaboration of Ullman's (1996, 1989) work on visual routines. Roelfsema's theory posits a base representation provided by early vision, a distributed process that generates low-level features such as edges and colored regions. At an intermediate level, so-called elemental operators, combined in visual routines, transform information from the base representation into an intermediate representation that captures spatial relationships and object features. Roelfsema (2005) identifies two main classes of elemental operators. Binding operators determine groupings among visual objects that are not established in early visual processing. This core set includes operators for searching, cuing, tracing, region filling, and association. Maintenance matching operators record intermediate results for further processing. These operators are supported in SegMan as well.

Cuing and searching: Spatial cuing involves the identification of properties of the visual pattern at a specified location. In SegMan, locations are screen coordinates, specified either locally (with respect to a given location) or globally. Visual patterns are described in terms of constraints on the values of region/group properties; these are accessible in declarative form as library templates. Pixel region properties include the number of pixels, the height and width, the area and coordinates of the region's bounding box, the location, and the red, green, and blue components of the region's color. Relational region properties include above/below, left/right, inside/outside, overlapping boundaries, or orientation at specific angles.

Visual search involves determining the location of a pre-defined visual pattern. In SegMan, search is based on the same set of library templates as with cuing; it involves traversing the set of pixel regions and groups corresponding to the current scene and identifying those from a specific template.

These two elemental operators are combined to support a simple approach to focusing attention. Searching or cuing can be limited to a specific area of the display, effectively providing a rectangular or circular pseudo-fovea. Within this area, pixels are processed exhaustively, while outside the region, information about sampled pixels remains available but is not processed into pixel regions.

Tracing and region filling: Curve tracing involves iteratively identifying connectivity properties between simple visual elements. Region filling can be viewed as a generalization of curve tracing to two dimensions, by identifying areas to be characterized as single visual objects. In SegMan, region filling is performed automatically during the segmentation stage. Tracing is handled as a specialization of region filling, in which regions are tested for the extent to which they approximate lines.

For this set of binding operators, SegMan provides variant operators, which have been described in alternative approaches to representing elemental operators other than Roelfsema's theory (Chapman, 1992; Halelamien, 2004):

- Ray projection: SegMan can cast a ray, a straight line emanating from a specific location, in any direction, and can identify discontinuities (i.e., changes in pixel region membership) along the path.

- Boundary tracing: Once an object has been identified as a pixel region, its boundary with other regions can be walked for further processing.
- Line segment identification: One of the limitations of the pixel-based segmentation process in SegMan is that an arbitrarily arranged set of pixels, if connected as neighbors, will be identified as a single object, however complex its resulting shape (e.g., a continuous maze would be characterized as a single object). In some domains, line segments are of particular interest. SegMan can identify such segments as pixel regions of limited width and restricted curvature, even if the segments are part of more complex pixel regions.

Association: Associations are relationships between co-occurring features. SegMan supports associations in the form of the templates in its library as logical propositions.

Matching: In general, matching is the process of identifying similarities (or differences) between stimuli. In SegMan, matching is closely related to the class of binding operators. Matching is implicit in the types of constraints present in library templates. SegMan supports exact matching and comparisons between the numerical values of region/group properties and constants; user-defined matching functions can also be defined. An additional, specialized matching operator is supported for tracking moving objects.

Tracking moving objects: SegMan reprocesses information iteratively, and it maintains a representation of its immediately preceding results. Where SegMan's processing cycle is short enough that moving objects remain close to their previous positions, they can be identified as the same objects. Objects in successive passes are matched in a greedy optimization process.

SegMan approximates Roelfsema's (2005) theory. There are two important conceptual differences between the theory and the implementation: in representation and in processing. First and most obvious is the non-cognitive base representation in SegMan. Roelfsema's theory is based on an interpretation of neural activation patterns; we do not make theoretical claims about this level of processing in SegMan. As with most approaches to computational cognitive modeling, we first aim for faithfulness at a functional or task level (Marr, 1982). Second, SegMan has no sharp distinction between early and intermediate visual processing. Identification of edges, for example, is possible only at the intermediate level, through identification of boundaries between adjacent pixel regions. While SegMan supports the elemental operators and visual routines of Roelfsema's theory to the extent described above, the correspondence does not hold for lower-level processing. Nevertheless, SegMan provides a plausible set of intermediate-level visual operators and result types in its representation from the perspective of a cognitive model that uses SegMan as a substrate for interacting. The implementation of this level in a functioning software architecture is not part of the theory of vision, just as how rules are implemented are not part of Soar's theory (Newell, 1990, e.g., pp. 223, 486, & 253).

Exploring this Approach in an Example HRI task

SegMan can be evaluated in two ways. One way is to evaluate the model as a functional approach, that is, can it perform the task of interest using psychologically plausible mechanisms? Generally, we believe so, as we have used SegMan in combination with ACT-R to model a variety of tasks noted above. And, is the system's performance adequate for dynamic tasks? We have used SegMan with ACT-R to model performance in a driving task with many similarities to HRI to make suggestions for interface design (Ritter, Van Rooy, St. Amant, & Simpson, 2006). One practical question that arises for HRI is an extension of the questions here: Can the kind of information required of interaction for common HRI tasks, such as teleoperation, be converted from the way it is presented to users to a form appropriate for input to a cognitive model? With this study, which we present next, we can answer this question affirmatively.

A second way the model can be evaluated, common in cognitive modeling research, is using performance measures such as the duration of actions and the occurrence of errors. This approach has been used to develop the model presented here, using an iterative modeling process, trying to show why to take the model seriously as well as to show where it can be improved (Grant, 1962). In an initial implementation of the model, a trace-based protocol analysis (Ritter & Larkin, 1994) found that the initial model was much slower than users, largely due to the cognitive limitations imposed by the ACT-R architecture on perceptual and motor processing. One source for this discrepancy was the serial alternation between visual processing of the camera view and the navigation buttons; humans can generally exploit unchanging regions of their visual field to remember the locations of the objects. We thus incorporated a pseudo-fovea that limited visual processing to the camera view in the revised model and also had the model retrieve the locations of navigation buttons from memory.

We carried out both an evaluative and functional test of this combined architecture of SegMan and ACT-R as part of a larger HRI study (Kukreja, 2004). The results of the study we present here are used to provide evidence that the model can be taken seriously. It provides new behavior; it makes predictions accurate enough for engineering work; and the results are used to reflect on where the model could be improved.

ACT-R

The model developed here is built in ACT-R[1] (Anderson & Lebiere, 1998), a cognitive architecture that supports problem solving over a general range of task domains. Though other cognitive architectures such as EPIC (Kieras, Wood, & Meyer, 1997) and Soar (Newell, 1990) are used for evaluating interactive systems, ACT-R is in widest use. ACT-R is well suited for modelling human performance in many types of interactive computing tasks. It has been used, for example, to model tasks in driving (Salvucci, 2001), aviation (Byrne & Kirlik, 2005), and cell phone usage (St. Amant, Horton, & Ritter, in press). (More examples are available at act.psy.cmu.edu and in Gray, 2007.) ACT-R provides a general cognitive architecture well suited to modeling human behavior.

ACT-R models simulate internal cognitive processing, such as changes of attention and memory retrievals, as well as external actions, such as movement of the hand and fingers, producing changes in the state of the model and durations for the actions. Each cognitive step, such as retrieving stored information or moving attention to a specific location in the field of view, has a latency associated with it. Latencies and other model parameters are determined to be consistent with the psychological literature via experimental comparison and validation. Part of the philosophy behind the development of general cognitive architectures is that once models within the architecture have been validated over a sufficiently large number of tasks, it becomes unnecessary to repeat the validation process for existing parameters for new tasks.

Briefly, the ACT-R architecture is a production system theory of how human cognition works. There are two memory modules in ACT-R: declarative memory and procedural memory. Declarative memory stores facts, such as "a red object is the target", in the form of memory chunks. Procedural memory stores knowledge of behavioral components in the form of productions, such as "To turn left when driving, turn the steering wheel to the left". For each module, a dedicated buffer serves as an interface with that module to other modules. Perceptual and motor modules provide the equivalent of eyes and hands to the model. With the perceptual module, the model can “see” properties of objects in the environment; productions sent to the visual buffer can direct attention to specific objects

[1] ACT-R is an acronym that stands for Atomic Components of Thought-Rational.

and create chunks in declarative memory representing their properties. Production rules can then direct the motor module to manipulate such objects.

HRI user model

To apply this combined architecture to the HRI task, we also used a simple ACT-R 5 model to perform the task. The model performs the same driving task as humans do (and two other tasks using the robot programming interface, Kukreja, 2004). It uses 8 rules and 9 declarative memory chunks to drive. Sub-symbolic computations of knowledge utility were included that provided simple learning. The model code, example traces, and an example movie of the model driving the robot on a demo task is available at http://acs.ist.psu.edu/papers/ritterKSA07/

The model uses a pseudo-fovea focused on the camera view window. A pixel region template was defined to identify the path as a visual object in the camera view. If the center of the path object is within a fixed distance from the center of the camera view, the mouse pointer is moved to the forward navigation button, which is pressed for a fixed duration of approximately 135 ms. If the path is to the right or left, the appropriate navigation button is chosen and pressed, again for a fixed duration of approximately 35 ms. After picking up the cup it turns for 8 s (which also indicates the robot's turning speed). With delays in the system, both actions led to longer keypresses. If the path is not found, the model issues commands to move the robot to the right or left, randomly selected, using the navigation buttons. The model picks up the cup using a comparable strategy, with one difference: the model relies on the built-in ER1 object recognition process rather than its internal templates to identify the cup. It is possible to design a template in SegMan for such a pattern, but it is also reasonable to expect a model to interact with mixed sources of information in an HRI interface. Recognition for the cup-grasping task entails that SegMan identify a blue rectangle that the ER1 imposes on the camera image of the cup, rather than the cup itself.

Humans have some further forms of parallel processing in cognition, visual and motor processing (Gray, John, & Atwood, 1993; Meyer & Kieras, 1997). While such parallelism is supported between the relevant modules in ACT-R, its use in the initial model was not sufficient to match the speed of human users. In the revised model, processing in the motor and visual modules is bypassed in favor of direct access to primitive operators in SegMan. These revisions had the effect of bringing the model's predictions closer to human performance by significantly reducing its real-time processing. This approach can be useful when modeling perceptual-motor control tasks in

ACT-R (e.g., Ritter, Van Rooy, St. Amant, & Simpson, 2006), but there are more advanced ways of using ACT-R that let the model interact more quickly (Salvucci, personal communication, March 2006).

HRI study

The ER1 Personal Robot System, shown in Figure 2, is a simple robot from Evolution Robotics. The dimensions of the 3-wheeled ER1 are 24 x 16 x 15 inches, and it weighs about 10 pounds when equipped with a laptop, its standard configuration. The laptop, running Microsoft Windows, is responsible for onboard computing. The ER1 is an entry-level robot lacking advanced capabilities, but it has many features analogous to those in more expensive field robots and can perform some comparable tasks.

< Insert Figure 2 about here. >

Another laptop, which communicates via a wireless Internet connection with the onboard laptop, allows teleoperation of the ER1. Teleoperation is supported by an application included with the ER1 called the ER1 Control Center, as shown in Figure 3. On the upper left of the control center is the camera view of the ER1. In the lower left corner are four navigation buttons (triangles) and a control for the robot's gripper. The user presses the "up" arrow icon with the mouse to cause the robot to move forward; moving right, left, and in reverse are handled analogously. If the user releases the mouse button, the robot stops. As the robot moves, the camera view is updated automatically. The remaining area of the control center is devoted to programming the robot's behaviours.

Our study involved thirty users who carried out this task. Half of these were women and half men. They were all students between the age of eighteen and twenty-five. The users were asked to perform three tasks with the robot (navigate the robot to pick up the cup and return, and program it to do two tasks with its visual programming language). They did each of these tasks once, in a fixed order, and took between 30-45 minutes to complete these tasks. The participants were paid $US 7 each as compensation. All participants were extensively computer literate and were comfortable working in a Windows environment, but none had worked with this robot before and none had extensive experience working with robots or teleoperation. The users were given instructions about each task they had to perform.

A simple teleoperation task was defined for the environment shown in Figure 4. The user (human or model) drove the robot, using the navigation buttons in the ER1 control center, along a U-shaped path (with segments of

2.0 m, 1.7 m, and 1.82 m) from the home position to the end, picked up a cup, and drove back again. The three-segment path is shown in the camera view in Figure 3. At the midpoint of the task, once the robot has traversed the path in one direction, the user had to guide the robot into position to pick up a cup with its gripper, again using the buttons in the control center. Once the user guided the robot to the home position carrying the cup, the task ended. Both the ACT-R model and the human users' behavior was recorded using the RUI keystroke logger (Kukreja, Stevenson, & Ritter, in press).

< Insert Figure 3 & 4 about here. >

To explore the effects of seeing the robot and having seen the robot on navigation, users were divided into three groups of ten for the navigation task. Users in the *visible* condition were able to see the robot as they performed the teleoperation task; users in the *previously seen* condition saw the robot beforehand but not during the teleoperation task; users in the *unseen* condition did not see the robot before or during the task. We were not sure when we started out how well the model could perform the task, or how well the users would be able to perform that task. While our model does not take advantage of these conditions, these data will inform later models and system designers.

At the start of the navigation task, the robot was placed in the adjacent room and depending on the group the subject belonged to, they were able to look into that room or not. After the navigation task, the robot was brought into the room where the subjects could see it and program it. At the end of the tasks they were paid, thanked, and debriefed.

Results

The model's results were taken from four trials, and the time to move the robot through the course is used for comparison. The variance in the model's performance is relatively low as shown in the analyses that follow. The time predicted by ACT-R alone is not accurate in this task because ACT-R does not model the system response time (i.e., the time for the robot to move and turn). Cognition made up about 25% of the total time.

All subjects completed the tasks, but the data from five subjects was unusable. Two users in the Visible condition and two in the previously Seen condition were discarded because the course was later changed to support the model's performance, and one subject in the Visible was discarded as an outlier because it was more than 4 standard deviations slower.

Because the model performs the task, it predicts users' knowledge and strategies, and shows that the mechanisms proposed are sufficient to perform the task. Figure 5 shows the performance of the humans and the model (over four trials), with standard deviations shown with the error bars. Figure 5 shows that the time predictions and their variance are very comparable to human behavior on this task. In this preliminary comparison, the model's performance is not different from any of the groups of subjects using inferential statistics. The unseen and visible conditions are, however, reliably different, t(15, two-tailed) = 2.54, $p<0.05$).

< Insert Figure 5 about here. >

We also examined the time to drive to the cup and the time to drive back. Without learning, we would expect the time to be about the same between the two halves, but the model does have stochastic elements in it, so there will be variance, and there is the variance that occurs because of interacting with the real world, such as different amounts of traction and different battery voltages driving the motor. The time out and back would be important if there was learning in the human subjects. The plot in Figure 6 shows that initial time to do each half of the task was different across groups, that learning occurred in the human subjects, and that with learning the groups became more similar in time. On average, the model also improved its driving—the model was about 24 s faster on average coming back. This learning effect appears to be due to an outlier, because the model did not include a substantial learning component, and the outlier appears to represent a very fortuitous run where the first half was unusually slow (285 s) and the second half unusually fast (133 s). This is an illustration of the unanticipated effects that arise in physically realized robots and that might not exist in all software simulations of robots.

< Insert Figure 6 about here. >

Figure 7 shows the number of mouse clicks used by the human drivers in each group and by the model. The results show that the model has less variance than the humans do. This might not be surprising given that the model's results represent repeatedly sampling from the same individual, rather than across individuals. This result suggests that we have a population of users, and that the model only models one type of user, that is, the model's number of clicks is similar to some users, but does not have enough variance to model the distribution of our users.

< Insert Figure 7 about here. >

Longer mouse clicks hold down the turn or move keys longer, leading to longer turns and moves. Figure 8 shows the average time that the mouse was held down. The graph shows that the model was pretty consistent in how long it held down the mouse to turn or move, basically, a fixed time with some variance arising from the long turn at the mid-point. Subjects, particularly the Unseen condition, used a longer time (turning more) and a wider variety of times. This suggests that the model could be improved by making the mouse clicks be more responsive to the size of the deviation from the path and possibly other factors.

< Insert Figure 8 about here. >

The model provides a reasonable approximation of human performance. We know that this is a feasible task for humans because the model can perform the task. We know that it is a reasonable approximation because in most measures (total time, time by half, number of mouse clicks) the model is well within the model validity criterion recommended by John and Newell (1989) for using models as reasonable approximations of human performance if outputs do not deviate by more than 20% from observed data. In one measure, the average length of mouse clicks, the model is not within 20%, but we know why—now that we know the distribution, the model can be corrected.

Note that this model does not account for the differences between the conditions in the human navigation experiment. We used the different conditions to explore performance in this task, and it suggests that the model most closely approximates users who are in the Seen condition. To account for these differences between conditions we would have to expand the model in several ways, to include the ability to look at another view of the robot (of it driving as well as through its camera), and we would have to include a richer representation of the model itself, both of which are interesting, but considerable effort.

While the model's predictions for driving time and the human data compare reasonably well, there are some significant limitations on the model as a representation of human performance. We note the most important limitations here.

- Model development and veridicality. As mentioned above, the revised model was not tailored to the differences between the previously seen, unseen, and visible conditions for users. Further, the model was developed to match human performance and was not developed a priori. While this is common practice in cognitive model development in new areas, it is not desirable. Further experimentation is needed to validate the changes between the initial and revised models and to generate a model to create zero parameter predictions.

- Environmental constraints. The navigation path followed by users and the model, without obstructions or noise in interpretation, constitutes an environment that is basically static, deterministic, and largely predictable. While perception of the environment changes in response to the behavior of the robot, these changes in the environment are limited. With improvements in SegMan, including more robust pattern recognition, the model could be extended to more complicated environments, but this remains an area of active research, and will require more complex models.
- Task constraints. The navigation task, though common in HRI, takes a very simple form in this study. Navigation in more realistic environments involves dealing with obstacles, maintaining information about location, searching for alternative paths, and computing progress toward goals, none of which is required in the study task.
- Model constraints. In human performance, memory can substitute for visual search. This factor is reflected in the caching of button locations but not in path-following. If users steer too fast along turns and the path disappears from the camera view, they can remember the last seen location of the path and steer in the appropriate direction. The model resorts to random search when the path is no longer visible. More generally, the model, as a preliminary model, does not include any reasoning capabilities, aside from a basic control loop.

Given all these caveats, our work still leads to recommendations that we take up in the conclusions.

Conclusion

We have demonstrated that a low level cognitive user model can interact with an HRI interface directly to generate behavior predictions. The model, based on ACT-R and the SegMan perceptual-motor extension, interacts with an uninstrumented commercial off-the-shelf (COTS) interface to teleoperate a simple robot in a simple physical environment. The user model does this in real time, at about the same rate and with the same variance that people perform the task as measured by task time, learning, and by mouse actions.

Despite the preliminary nature of this experiment, this is to our knowledge the first instance of a low level cognitive model representing human information processing acting as a surrogate user in an HRI environment using the same user interface that human operators use. It extends the lessons for HRI from having a similar model drive an uninstrumented driving game (Ritter, Van Rooy, St. Amant, & Simpson, 2006), but in this case it drives a real robot using the video display in the HR interface.

This result is significant for a few reasons. First, it shows that real-time performance is possible for a cognitive model interacting with a real rather than a simulated environment, with performance that is consistent with human behavior to a good approximation. Similar work has not used the same interface, has not interacted with real robots, or has not made timing predictions. Second, it shows that the approach we have taken can handle mixed visual sources, physical environment information from a camera as well as virtual information from a graphical user interface, in a single framework. These are limited observations and simply a modest first step for the study of HRI with cognitive models, but this result still advances the state of the art, and shows that this approach can be applied to a wider range of interfaces and tasks. This result makes suggestions for the development of its components, for HRI design. We discuss these, and then consider the most salient limitations and future work.

Recommendations for SegMan and ACT-R

This work makes recommendations for the development of ACT-R, SegMan, and cognitive architectures in general. ACT-R's cognitive components did not have to be modified, but they were used to model a fairly simple task in a fairly simple way. The use of SegMan should be seen as an extension to ACT-R to support direct interaction with existing interfaces. Future work with interaction will need to use more sophisticated reasoning strategies and should be able to use the spatial reasoning modules and models being developed in ACT-R (Gunzelmann & Anderson, 2002; Harrison & Schunn, 2003; Johnson, Wang, & Zhang, 2003).

One modification to SegMan was necessary to support interacting with this more complex task, and this modification has theoretical implications for similar engineering theories of vision and provides a useful argument about why human vision is the way it is. SegMan was modified using existing parameters to look at a subsection of the screen, creating a virtual fovea. This made the theory more representative of how human vision is performed and made the visual processing faster.

The project provides evidence that unified theories of cognition are not made up solely from task knowledge and a cognitive architecture to apply the knowledge. They need to be integrated with their task environment (Ritter, Baxter, Jones, & Young, 2000). Interaction mechanisms to provide information to the cognitive architecture and to provide a way for knowledge to be applied to the world are very real aspects of this task and are necessary for allowing the model to act. The interaction mechanisms are separate from the cognitive mechanisms. They are implemented in different modules, and these modules have different characteristics from the cognitive modules—

they live closer to the environment and use different representations. This work suggests that unified theories of cognition might be better decomposed into knowledge plus cognitive mechanisms plus interaction mechanisms.

Recommendations for HRI interface design

The model gives insights about how to improve the ER1 interface. Studies in cognitive modelling such as the one in the previous section would ideally point to improvements in the user interface. We noted above a few of the difficulties the model has using the interface. These difficulties included using the small buttons, and inherently the troubles driving using multiple mouse clicks. These analyses could be used to show HRI designers where people would have difficulties as well. Unfortunately, our work has not yet progressed to the point where it is possible to identify interaction problems that cannot be identified by more conventional evaluation methods such as inspection or simple user testing. These results might be more convincing, however, as the analysis is based on a theoretical analysis that can be used to perform the task, and the implications can perhaps be seen more directly in the interface, and there is the potential to do these analyses automatically.

Limitations, implications, and future work

There remain practical problems with using the approach routinely. First, many more tasks will have to be modelled before this approach can be used routinely for evaluation. Second, using models with robots has technological issues that HCI does not have. As with the ACT-R architecture, SegMan allows durations to be specified for the execution of its operators that are independent of their actual real-time duration. Because this model interacts with hardware, making time predictions is more difficult to record accurately because the model's timing has to be synchronized with and interact with the robot platform, and it has to run in real-time. Third, the factors that are most relevant to evaluation of interactive systems tend to deal with information processing at the cognitive rather than the perceptual level. In this task, the importance of interaction and cognition are more equally balanced.

Nevertheless, this demonstration of the feasibility of applying cognitive modeling techniques to an HRI task provides a theory of how the interface is used, and the theory can start to make quantitative predictions useful for analysing tradeoffs in design. Models like this will be able to test the interface for completeness in ways that more descriptive methods cannot. The models, once created, may be able to be modified to help users perform the task. And they will be able to help test the robots as well, because they can put the robot through its behaviors.

Acknowledgments

This project was sponsored by the National Science Foundation, awards IIS-0083281 and ITR-0426852, and by the Office of Naval Research, contract number N00014-02-1-0021. The study was performed at Penn State in the College of IST's Shared HCI Lab. Comments by Rajal Cohen, Sue Kase, Toby Mordkoff, and three anonymous reviewers have helped improve this report. Laura Klein helped with the study design and materials; Adrienne Bonski supported by a WISER fellowship helped maintain the robot. The views in this paper do not necessarily reflect the position or the policies of the U.S. Government, and no official endorsement should be inferred.

References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Byrne, M. D. (2001). ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies, 55*(1), 41-84.

Byrne, M. D., & Gray, W. D. (2003). Returning Human Factors to an engineering discipline: Expanding the science base through a new generation of quantitative methods. Preface to the Special Section. *Human Factors, 45*(1), 1-4.

Byrne, M. D., & Kirlik, A. (2005). Using computational cognitive modeling to diagnose possible sources of aviation error. *International Journal of Aviation Psychology, 15*(2), 135-155.

Casper, J., & Murphy, R. (2003). Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. *IEEE Transactions on Systems, Man, and Cybernetics Part B, 33*(3), 367-385.

Chapman, D. (1992). Intermediate Vision: Architecture, implementation, and use. *Cognitive Science, 16*(4), 491-537.

Cradall, J. W., Nielsen, C. W., & Goodrich, M. A. (2003). Towards predicting robot team performance. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics,* 906-911.

Drury, J. L., Hestand, D., Yanco, H. A., & Scholtz, J. (2004). Design guidelines for improved human-robot interaction. In *CHI Extended Abstracts 2004,* 1540. New York, NY: ACM.

Gluck, K. A., Ball, J. T., Krusmark, M. A., Rodgers, S. M., & Purtee, M. D. (2003). A computational process model of basic aircraft maneuvering. In *Proceedings of the Fifth International Conference on Cognitive Modelling,* 117-122. Bamberg, Germany: Universitäts-Verlag Bamberg.

Gluck, K. A., & Pew, R. W. (Eds.). (2005). *Modeling human behavior with integrated cognitive architectures: Comparison, evaluation, and validation*. Mahwah, NJ: Erlbaum.

Grant, D. A. (1962). Testing the null hypothesis and the strategy and tactics of investigating theoretical models. *Psychological Review, 69*(1), 54-61.

Gray, W. D. (Ed.). (2007). *Integrated models of cognitive systems*. New York: Oxford University Press.

Gray, W. D., John, B. E., & Atwood, M. E. (1993). Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world task performance. *Human-Computer Interaction, 8*(3), 237-309.

Gunzelmann, G., & Anderson, J. R. (2002). Strategic differences in the coordination of different views of space. In *Proceedings of the Twenty-Fourth Annual Conference of the Cognitive Science Society,* 387-392. Mahwah, NJ: Erlbaum.

Halelamien, N. S. (2004). *Visual routines for spatial cognition on a mobile robot.* M. S. thesis, Department of Computer Science, Carnegie-Mellon University.

Harrison, A. M., & Schunn, C. D. (2003). ACT-R/S: Look Ma, no "cognitive-map"! In *Proceedings of the Fifth International Conference on Cognitive Modelling,* 129-134. Bamberg, Germany: Universitäts-Verlag Bamberg.

John, B. E., & Newell, A. (1989). Cumulating the science of HCI: From S-R compatibility to transcription typing. In *Proceedings of CHI, 1989,* 109-114. New York, NY: ACM.

Johnson, T. R., Wang, H., & Zhang, J. (2003). An ACT-R model of human object-location memory. In *Proceedings of the 25th Annual Meeting of the Cognitive Science Society,* 1361. Mahwah, NJ: Erlbaum.

Kaber, D. B., Wang, X., & Kim, S.-H. (2006). Computational cognitive modeling of operator behavior in telerover navigation. In *Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics,* (CD-ROM). Taipei, Taiwan, October 8-11, 2006.

Kieras, D. E. (1985). The why, when, and how of cognitive simulation. *Behavior Research Methods, Instrumentation, and Computers, 17*, 279-285.

Kieras, D. E., Wood, S. D., Abotel, K., & Hornof, A. (1995). GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'95),* 91-100. New York, NY: ACM.

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1995). Predictive engineering models using the EPIC architecture for a high-performance task. In *Proceedings of the CHI '95 Conference on Human Factors in Computing Systems,* 11-18. New York, NY: ACM.

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1997). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *Transactions on Computer-Human Interaction, 4*(3), 230-275.

Kukreja, U. (2004). *Towards model-based evaluations of human-robot interfaces.* Unpublished MS thesis, Department of Computer Science and Engineering, The Pennsylvania State University.

Kukreja, U., Stevenson, W. E., & Ritter, F. E. (in press). RUI—Recording User Input from interfaces under Windows and Mac OS X. *Behavior Research Methods*.

Marble, J. L., Bruemmer, D. J., Few, D. A., & Dudenhoeffer, D. D. (2004). Evaluation of supervisory vs. peer-peer interaction with human-robot teams. In *HICSS 2004,* 50130b.

Marr, D. (1982). *Vision*. San Francisco: W. H. Freeman Press.

Meyer, D. E., & Kieras, D. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review, 104*(1), 3-65.

Murphy, R. R., Blitch, J., & Casper, J. (2002). AAAI/RoboCup-2001 Urban Search and Rescue Events: Reality and competition. *AI Magazine, 23*(1), 37-42.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Nielsen, J. (1993). *Usability engineering*. Chestnut Hill, MA: AP Professional Press.

Olsen Jr., D. R., & Goodrich, M. A. (2003). Metrics for evaluating human-robot interactions. In *Proceedings of PERMIS 2003*: NIST Special Publication 1014. www.isd.mel.nist.gov/research_areas/research_engineering/Performance_Metrics/PerMIS_2003/Proceedings/.

Ritter, F. E. (2004). Choosing and getting started with a cognitive architecture to test and use human-machine interfaces. *MMI-Interaktiv-Journal, 7*, 17-37. useworld.net/mmiij/musimms.

Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2000). Supporting cognitive models as users. *ACM Transactions on Computer-Human Interaction, 7*(2), 141-173.

Ritter, F. E., & Larkin, J. H. (1994). Using process models to summarize sequences of human actions. *Human-Computer Interaction, 9*(3), 345-383.

Ritter, F. E., & Major, N. P. (1995). Useful mechanisms for developing simulations for cognitive models. *AISB Quarterly, 91*(Spring), 7-18.

Ritter, F. E., Van Rooy, D., & St. Amant, R. (2002). A user modeling design tool based on a cognitive architecture for comparing interfaces. In *Computer-Aided Design of User Interfaces III, Proceedings of the 4th International*

Conference on Computer-Aided Design of User Interfaces CADUI'2002, 111-118. Dordrecht, NL: Kluwer Academics Publisher.

Ritter, F. E., Van Rooy, D., St. Amant, R., & Simpson, K. (2006). Providing user models direct access to interfaces: An exploratory study of a simple interface with implications for HRI and HCI. *IEEE Transactions on System, Man and Cybernetics, Part A: Systems and Humans, 36*(3), 592-601.

Roelfsema, P. R. (2005). Elemental operations in vision. *Trends in Cognitive Sciences, 9*(5), 226-233.

Russell, S., & Norvig, P. (1995). *AI: A modern approach*. Prentice-Hall: Englewood Cliffs, NJ.

Salvucci, D. (2001). Predicting the effects of in-car interface use on driver performance: An integrated model approach. *International Journal of Human-Computer Studies, 55*(1), 85-107.

Schoelles, M. J., & Gray, W. D. (2001). Argus: A suite of tools for research in complex cognition. *Behavior Research Methods, Instruments, & Computers, 33*(2), 130-140.

Scholtz, J. (2002). Evaluation methods for human-system performance of intelligent systems. In *Proceedings of the 2002 Performance Metrics for Intelligent Systems (PerMIS) Workshop*.

Scholtz, J. (2003). Theory and evaluation of human robot interactions. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on System Sciences,* p. 125a. 110.1109/HICSS.2003.1173627. IEEE Computer Society.

Scholtz, J., Antonishek, B., & Young, J. (2003). Evaluation of operator interventions in autonomous off-road driving. In *Performance Metrics for Intelligent Systems (PerMIS),* Gaithersburg, MD: NIST.

Scholtz, J., Antonishek, B., & Young, J. (2004). Evaluation of a Human-Robot Interface: Development of a Situational Awareness Methodology. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 5,* 50130c. IEEE Computer Society.

St. Amant, R., Horton, T. E., & Ritter, F. E. (2004). Model-based evaluation of cell phone menu interaction. In *Proceedings of the CHI'04 Conference on Human Factors in Computer Systems,* 343-350. New York, NY: ACM.

St. Amant, R., Horton, T. E., & Ritter, F. E. (in press). Model-based evaluation of expert cell phone menu interaction. *Transactions on CHI.*

St. Amant, R., Riedel, M. O., Ritter, F. E., & Reifers, A. (2005). Image processing in cognitive models with SegMan. In *Proceedings of HCI International '05,* Volume 4 - Theories Models and Processes in HCI. Paper # 1869.

St. Amant, R., & Riedl, M. O. (2001). A perception/action substrate for cognitive modeling in HCI. *International Journal of Human-Computer Studies, 55*(1), 15-39.

Thimbleby, H. (2002). Reflections on symmetry. In *Proceedings of Advanced Visual Interfaces,* 28–33.

Trafton, J. G., Schultz, A. C., Perznowski, D., Bugajska, M. D., Adams, W., Cassimatis, N. L., & Brock, D. P. (2006). Children and robots learning to play hide and seek. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction,* 242-249. New York, NY: ACM Press.

Treisman, A. M., & Gelade, G. (1980). A feature integration theory of attention. *Cognitive Psychology, 12*, 97–136.

Ullman, S. (1989). Aligning pictorial descriptions: An approach to object recognition. *Cognition, 32*, 193-254.

Ullman, S. (1996). *High-level vision*. Cambridge, MA: MIT Press.

Urbas, L., & Leuchter, S. (2005). Model based analysis and design of human-machine dialogues through displays. *KI – Zeitschrift für künstliche Intelligenz [AI-Journal for AI]*, 45-51.

Yanco, H., A., Drury, J. L., & Scholtz, J. (2004). Beyond usability evaluation: Analysis of human-robot interaction at a major robotics competition. *Human-Computer Interaction, 19*(1&2), 117-149.

Figure Captions and Figures

Figure 1. A schematic of four pixel regions, with annotations.

Figure 2. An ER1 robot, configured its gripper in a high location (holding a cup), its camera (on the top of the stalk), and the interface that is mirrored on another computer (shown in Figure 3) that controls the robot remotely.

Figure 3. The HRI interface that users saw when driving the ER-1 robot remotely.

Figure 4. The physical task environment for the human users. The robot's path for human subjects was red on a blue-gray striped floor; the model drove on a table that provided greater image contrast using a black path on a white tabletop.

Figure 5. Comparison of the model's predicted task time with human performance. Standard deviations shown as error bars.

Figure 6. Comparison of learning on the predicted and actual task time for the model and three conditions. Standard deviations shown as error bars.

Figure 7. The number of mouse clicks (and standard deviations) by the human drivers in each group and by the model to navigate the course and pick up the cup.

Figure 8. Average mouse click duration for human drivers in each group and by the model.

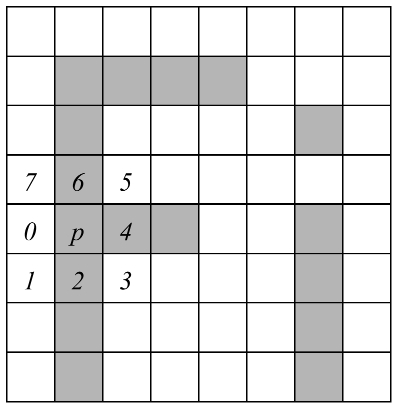


Figure 1. A schematic of four pixel regions, with annotations.

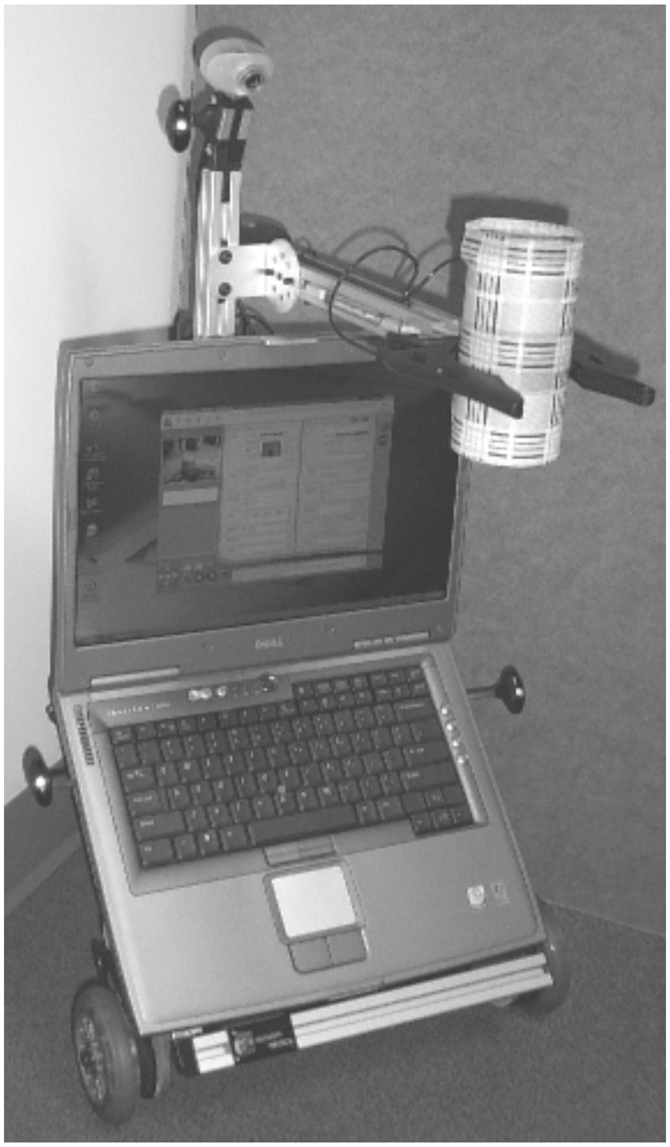
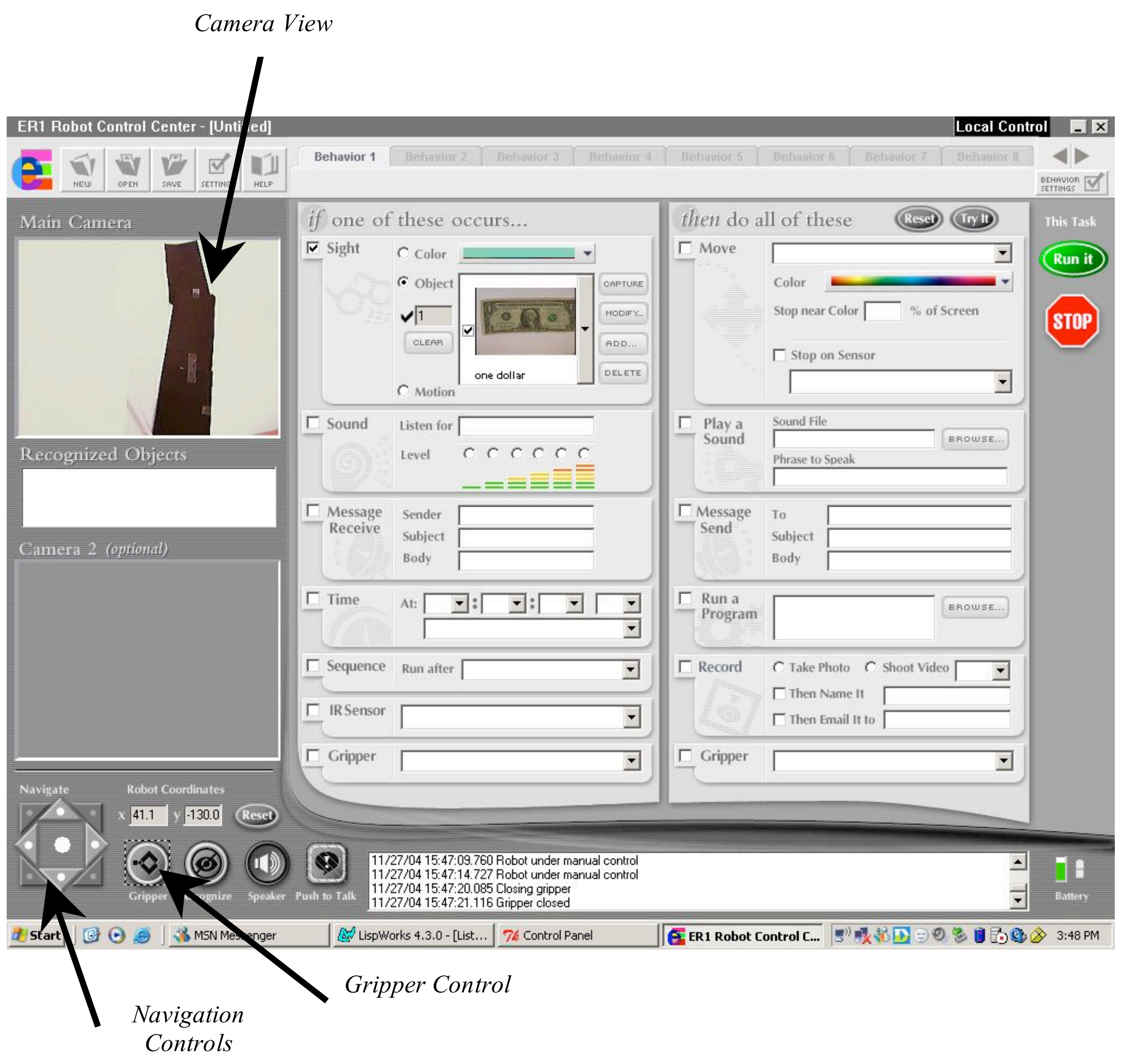


Figure 2. An ER1 robot, configured its gripper in a high location (holding a cup), its camera (on the top of the stalk), and the interface that is mirrored on another computer (shown in Figure 3) that controls the robot remotely.



```
11/27/04 15:47:09.760 Robot under manual control
11/27/04 15:47:14.727 Robot under manual control
11/27/04 15:47:20.085 Closing gripper
11/27/04 15:47:21.116 Gripper closed
```

Figure 3. The HRI interface that users saw when driving the ER-1 robot remotely.

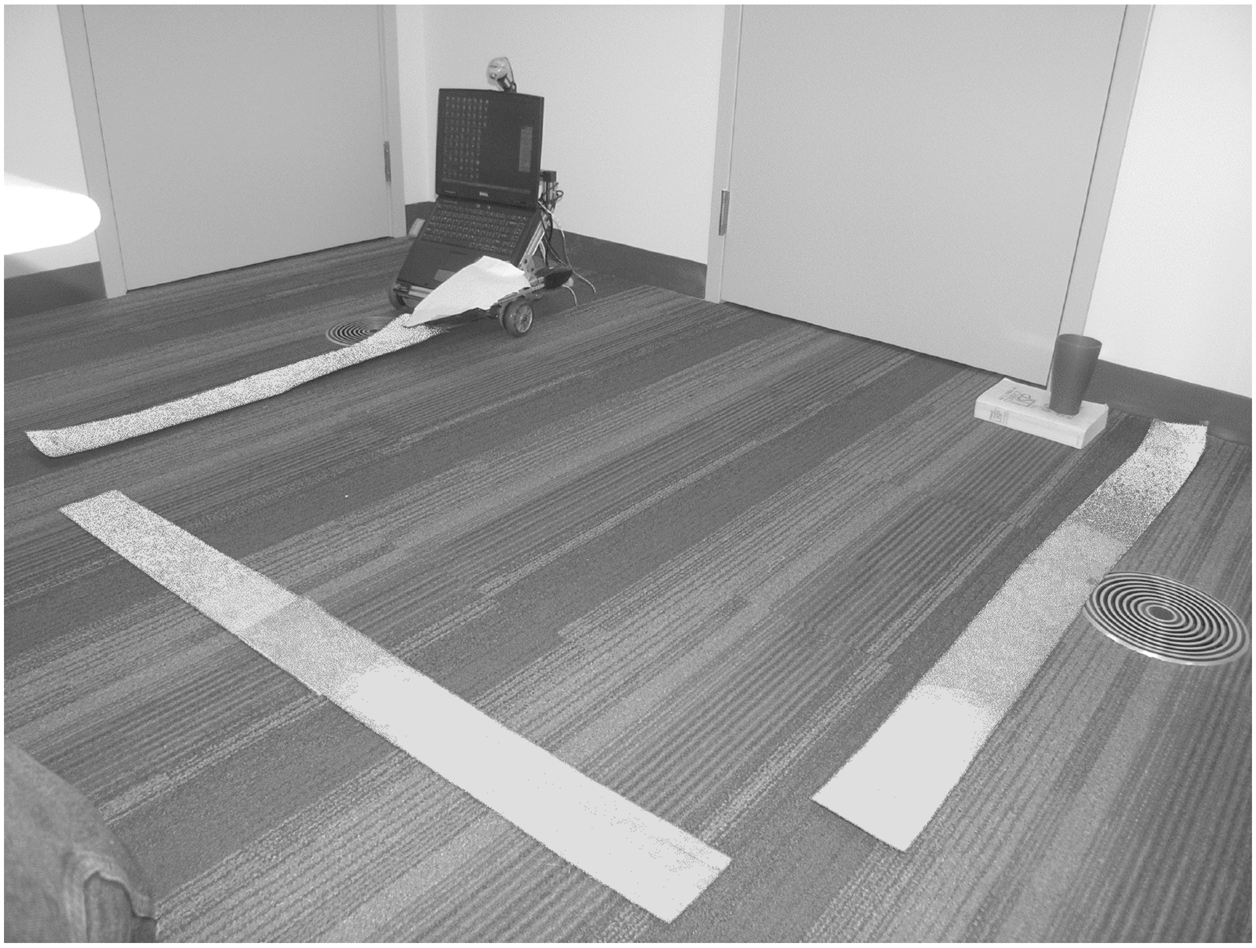


Figure 4. The physical task environment for the human users. The robot's path for human subjects was red on a blue-gray striped floor; the model drove on a table that provided greater image contrast using a black path on a white tabletop.

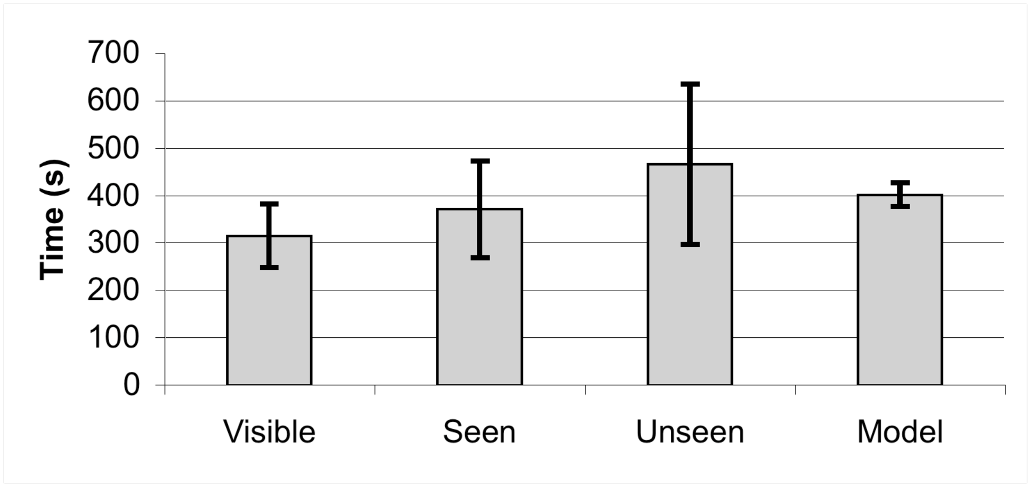


Figure 5. Comparison of the model's predicted task time with human performance. Standard deviations shown as error bars.

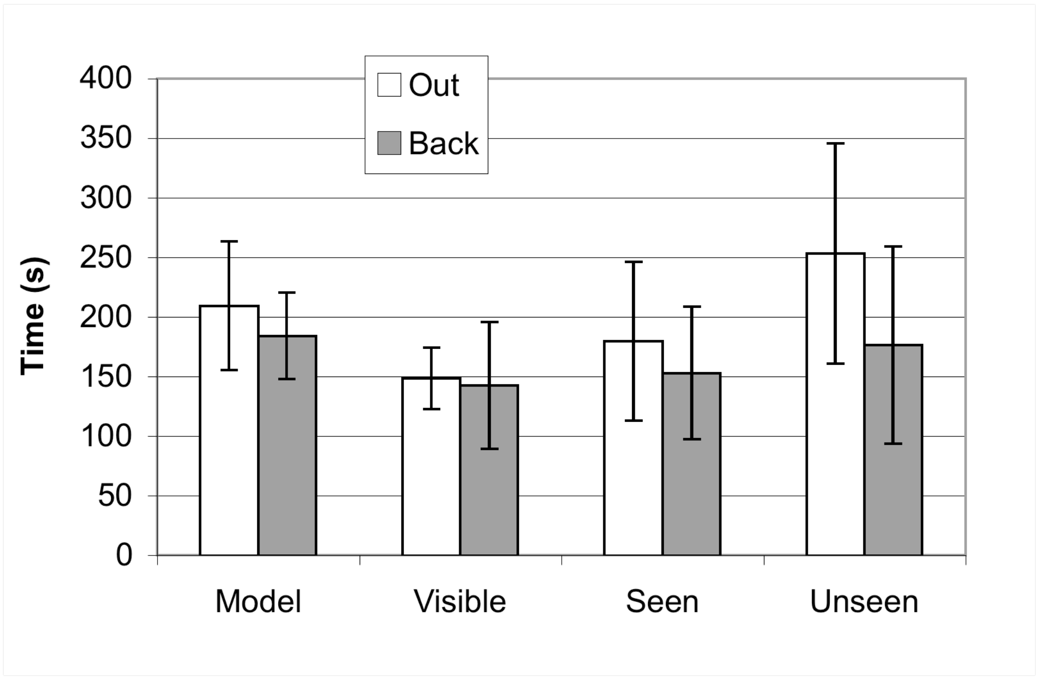


Figure 6. Comparison of learning on the predicted and actual task time for the model and three conditions. Standard deviations shown as error bars.

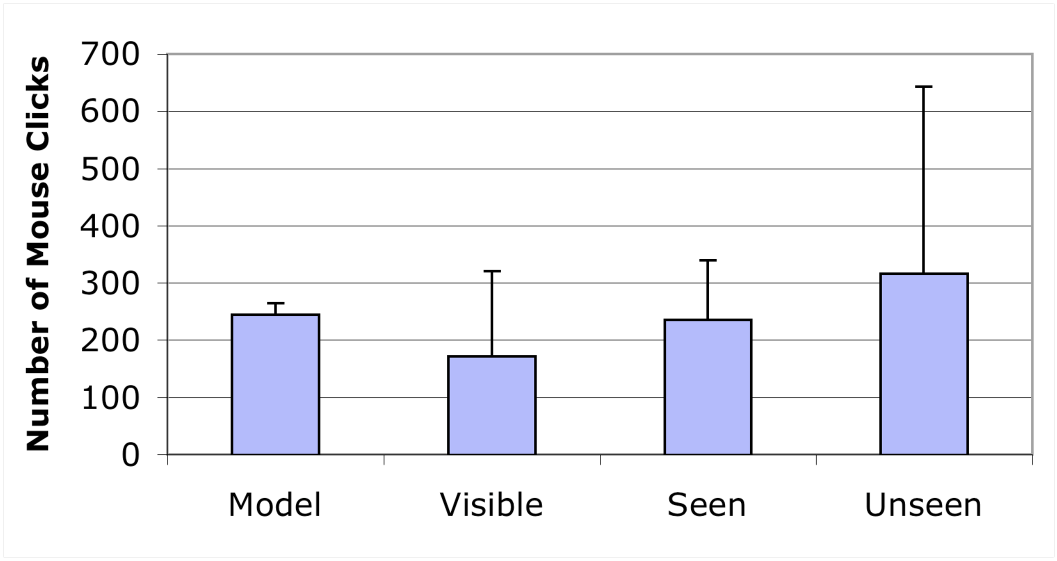


Figure 7. The number of mouse clicks (and standard deviations) by the human drivers in each group and by the model to navigate the course and pick up the cup.

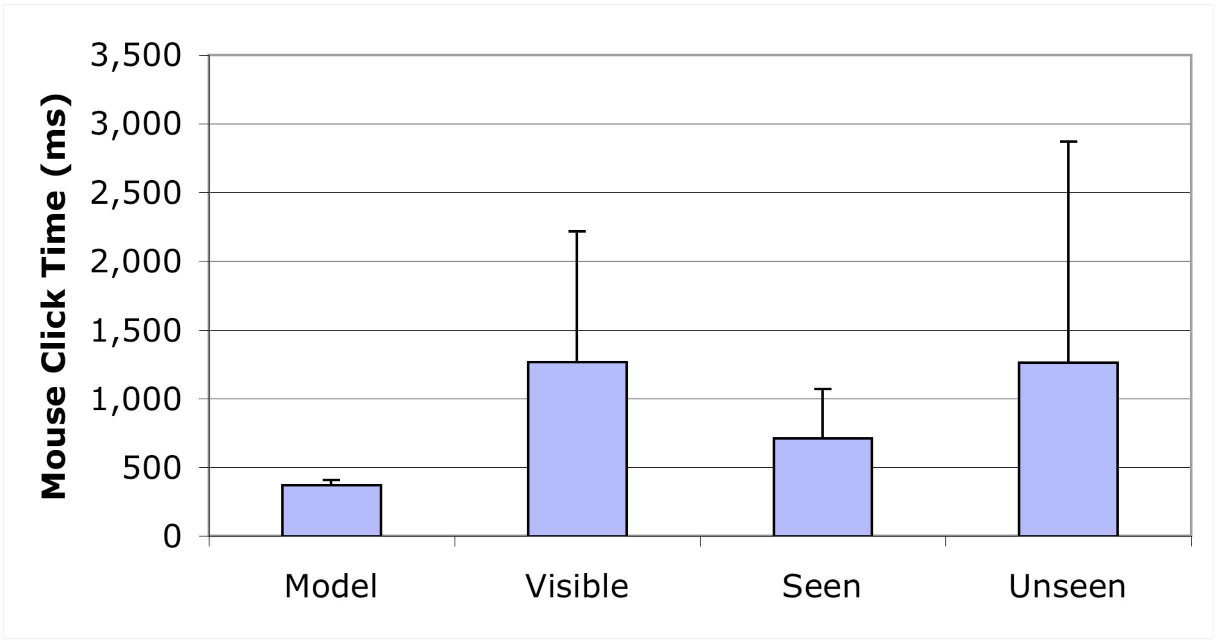


Figure 8. Average mouse click duration for human drivers in each group and by the model.